

システムの記事

低コスト・低消費電力のカーナビ・システムを構築するためのハード&ソフト設計

——拡張性のあるプロセッサ・ベースの開発プラットホーム作り

Paffen Matthijs

ここでは，カー・ナビゲーション・システムのプラットホーム基板の開発事例を紹介する．ここで取り上げるのは，低価格帯のシステムをターゲットとしたボードである．一つのプロセッサで測位や電子地図表示，音声ガイダンスなどの処理を行うためには，ソフトウェアを開発する際に各処理に必要なCPUパワーを考慮してタスクを振り分ける必要がある．（編集部）

以前は高級車だけに搭載されていたカー・ナビゲーション・システムですが，今や大衆車にまで急速に普及してきています．近い将来，快適な経路探索を提供するこうしたシステムは，すべての車に搭載されることになると思われます．あるいは，すべての自動車に搭載とまではいかないまでも，メーカ・オプションとして比較的低価格で提供されることになるでしょう．すでに価格対性能比に優れた製品も登場しており，こうした予測は現実のものとなりつつあります．

● 拡張性に対応するにはプラットホームが重要

ただし，国や地域ごとに使用する状況が異なるため，さまざまな要件がナビゲーション・システムに課せられることになります．これらの要件は，単に複数の言語や現地の地図を提供するということだけではありません．例えば，車が移動している間に新しい経路を選ぶことが法律で禁じられている国もあります．

こうした仕向け地仕様のカスタマイズ，および低価格帯から高価格帯までのシリーズ展開に対応するため，拡張性のある開発プラットホームが用意されるようになりました．開発プラットホームを用いることで，自動車メーカ（または車載機器メーカ）はエンド・ユーザに製品を届ける直前までアップデートを行えます．同時に，自動車メーカは一つのハードウェア・プラットホームだけを使用すればよいという利点を活用することで，開発，検証，保守にかかるコストを削減できます．なお，ナビゲーション・システムのプラットホームには，いくつかの最低限の機能を搭載しておく必要があります．例えば，位置を取得するためのGPS（Global Positioning System），ユーザ・インターフェースとなるキーパッドやディスプレイなどは備えておくべきでしょう．

音声や地図ガイダンス（矢印，矢印＋地図）によって運転者は目的地に誘導されますが，こうした地図情報や経路計算値をCD/DVD-ROMや半導体メモリ（フラッシュ・メモリなど）に格納するナビゲーション・システムを「オンボード型」と呼びます．現時点では，市場の主流はこのオンボード型です．一方，GPRS（general packet radio service）などの無線通信を利用してサーバに接続して情報をやりとりする「オフボード型ナビゲーション・システム」と呼ばれるものもあります．

開発コストを削減するには，記録媒体（DVD-ROM，ハード・ディスク装置，半導体メモリなど）やディスプレイ・サイズの選択も重要です．低価格帯のシステムでは，多くの場合，単純な音声と矢印ベースのナビゲーション（進路変更ごとの指示），小規模な地図を使用しています．高価格帯のシステムでは，2次元または3次元グラフィックスで地図を表示する5～8インチのカラー・ディスプレイを使用しています．

システムの拡張性をさらに向上させるには，コネクタを介してハードウェアの一部を接続・交換できるようにしたり，ソフトウェアの更新によってシステムを再構成できるようにすることも必要になります．

1．低価格帯カーナビのハードウェア構成

現在のナビゲーション・システムには，経路計算や音声ガイダンスのためのオーディオ・アルゴリズムなどのタスクを処理するDSP（ディジタル信号処理プロセッサ），GPSチップセットのタイミングと係数の更新といった標準的な制御タスクおよびGUIタスクを処理するマイクロコントローラが搭載されています．

最近では，DSPとマイクロコントローラの機能を1チップに集積したデバイスを用いるケースもあります．このようなデバイスの一つとして，米国Analog Devices社のBlackfinプロセッサがあります．このプロセッサは，動作周波数が最大600MHzで1サイクル当たり二つの積和演算（16ビット）を同時に実行します（**図1**）．また，PPI（parallel peripheral interface），SPORT（シリアル・ポート），外部パラレル・インターフェースなどの高帯域なインターフェースを備えています．DSPとしての機能はマイクロコントローラに組み込まれており，RISCプロセッサのプログラミング・モデルを利用しています．16ビットと32ビットの可変長命令により，コード密度が改善されるだけでなく，処理機能と並列してロード/ストア命令を発行することができます．

ここでは，このデバイスを用いたカー・ナビゲーション・システムのプラットホーム基板の開発例を紹介します（p.107のコラム「低価格帯カーナビのリファレンス設計」を参照）．

● 位置検出はGPSと推測航法を組み合わせて行う

一般的な低価格帯のカー・ナビゲーション・システムの例を**図2**に示します．この例では，搭載されているプロセッサは一つ（BlackfinのADSP-BF539）だけです．システムをリセットした後，ブート・プロセスが開始され，NOR型フラッシュ・メモリとの間のインターフェースであるSPI（serial peripheral interface）がマスタ・モードで使用されます．NOR型フラッシュ・メモリは，耐久性が10年という車載部品に対する品質基準を満たしつつ，極めて長いデータ保持期間を備えています．また，リニア・アドレス指定も可能になっています．メモリに格納されているナビゲーション用アプリケーション・ソフトウェアの容量はかなり大きいため，ブート後，合計32MバイトのSDRAMが必要となります．

ナビゲーション・システムには位置を検出する機能の搭載が必須ですが，これは一般的な組み込み技術で解決できます．**図2**の例では，GPSの各チャネル間の相関をとるためのLSI（200XST）を用いて，12個のGPS衛星を追跡・捕

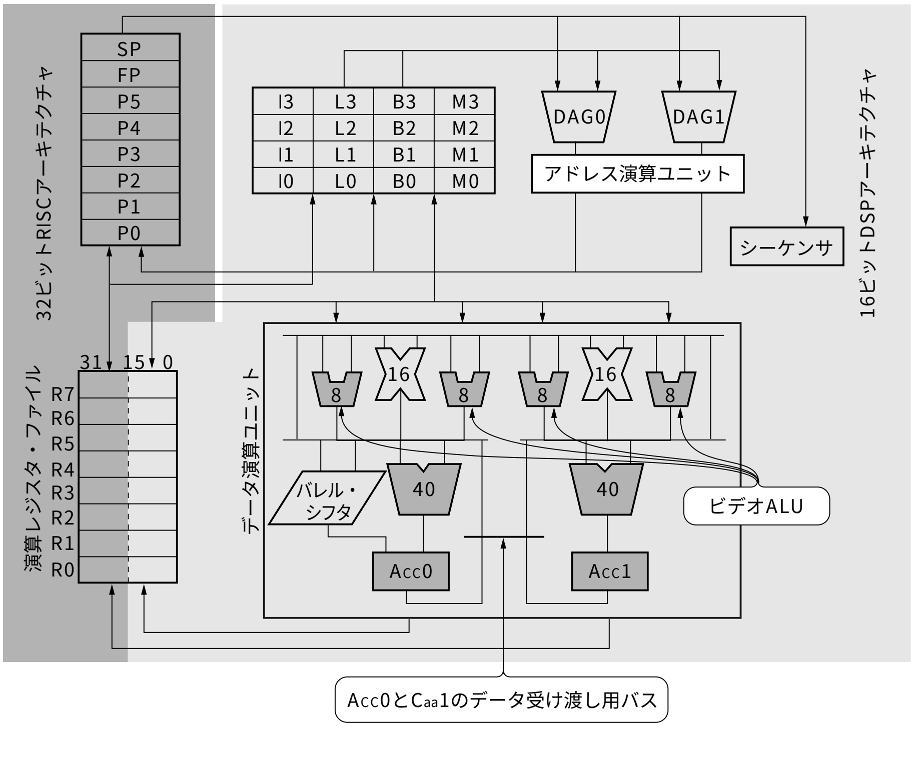


図1
DSPとマイクロコントローラを1チップに集積
Analog Devices社のBlackfinプロセッサのコア・アーキテクチャを示す．二つの積和演算ユニットや四つのビデオALUなどのDSP的なアーキテクチャに加え，SP（スタック・ポインタ）/FP（フレーム・ポインタ）などのRISC的なアーキテクチャも採用している．外部インターフェースとして，TWI（two-wire serialinterface；I²C互換）やCAN（controller area network），SPI（serial peripheral interface），UARTなどに対応している．

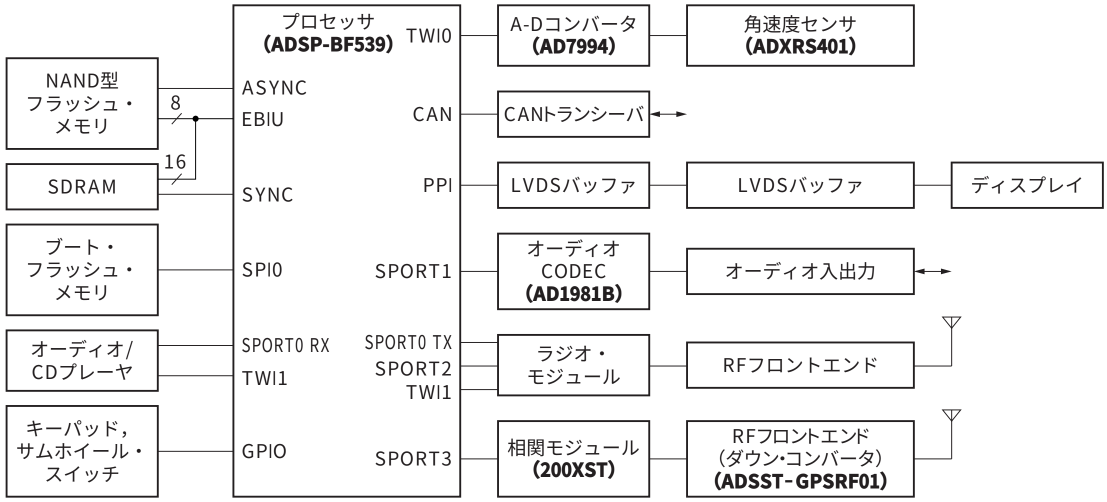


図2
ナビゲーション・システムのハードウェアのブロック図
ナビゲーション・システムの例として，Blackfin プロセッサ(ADSP-BF539)を用いた低価格帯のハードウェア構成を示す．

捉しています．これと同時に，プロセッサ上で制御および推定アルゴリズムを実行します(このときCPUパワーの約10%を消費する)．

RFフロントエンドLSIは，1.575GHzのL1バンド注1をミキシングして，相関ユニットに渡します．GPS衛星の信号が劣化しても位置を見失わないようにするため，GPS衛星からの信号とは別に，三つの信号を取得しています．この三つの信号を組み合わせる「推測航法」により，トンネルの中などでの自動車の位置精度が向上します．一つ目の信号は，角速度センサから得られる2次元の加速度(測定値)です．リアルタイム・クロックと元の方向をもとに，新しい方向を推測できます．二つ目と三つ目の信号はCANバス経由で得られる速度と方向(自動車が前進しているのか後進しているのか)に関する情報です．

一般に，推測航法とGPSを組み合わせて使用する場合，それぞれの情報に重み付け(指標)を施します．推測航法の場合は何度も急旋回したときは精度が低下しますが，GPSの場合はGDOP(幾何学的測定精度)に応じて位置精度が低下します．この指標は，衛星の数と衛星間の距離を考慮して，推定した位置の品質を表します．

● システムを構成するモジュールの役割

図2のシステムでは，電子地図の生のベクトル情報をCD-ROMに格納します．ほとんどの場合，1枚のCD-ROMに少なくとも欧州の一ヵ国分のベクトル情報を格納できます．

ナビゲーション中にCD-ROMなどの光学デバイスに何度もアクセスすることを防ぐため，32MバイトのNAND型フラッシュ・メモリが用意されています．このフラッシュ・メモリには，計算した経路(corridorとも呼ばれる)を格納します．格納された経路は複数の分岐を備えています．ユーザが最初に用意されていた経路からそれても，この分岐情報を利用すれば，経路の再計算を行わなくてすみます．

目的地の入力には，汎用I/O(GPIO)またはPF(programmable flag)ピン注2に接続されたキーパッドやサムホイール・スイッチを用います．また，目的地を音声入力で制御するには，オーディオCODECに接続されたマイクロホンを使用できます．この場合，システムに自動音声認識ソフトウェア・モジュールを搭載する必要があります．

ディスプレイは，進路変更ごとの情報または地図ベースの情報のいずれかを表示します．**図2**のシステムにおいて，ディスプレイは27MHzのクロックを必要とします．長いフラット・ワイヤ(パラレル)ケーブルを用いると，このクロックによって大きな電磁干渉(EMI：electro-magnetic interference)が発生します．そこで，図のシステムではLVDS(low voltage differential signaling)バッファを使って，EMC(electro magnetic compatibility；電磁環境適合性)の要件を満たしつつ，長いケーブルを使用できるようにしています．

渋滞の場合，ラジオからのRDS(radio data system)/RDBS(radio data broadcasting system)またはTMC(traffic message channel)などのメッセージによって，ナビゲーションの方向を変更します．ユーザには，ナビゲーション中に音声メッセージ(オーディオCODECを介して)やラ

注1：L1バンドとは，現在，民間用GPSに割り当てられている周波数(1.57542GHz)のこと．
注2：このI/Oは割り込み入力として使用できる．

Column1 低価格帯カーナビのリファレンス設計

現在，筆者らは低価格帯のカー・ナビゲーション・システムのリファレンス設計を開発しています(**写真A-1**).

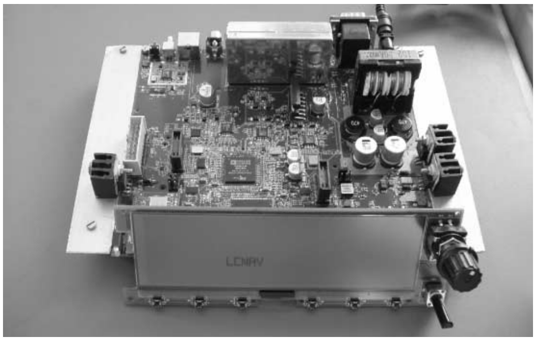
このシステムは，デモンストレーションを目的としたものであると同時に，機器メーカがすばやく設計を開始し，個々の要件に合わせてこれをカスタマイズできるというコンセプトを実現させるものです．一般的なシステムと比べてより多くのハードウェア機能が利用可能であり，高い自由度を備えています．

最新バージョンでは，本文の**図2**で示したシステムに次のような機能を追加しています．

- Bluetoothトランシーバ
- アナログ・ディスプレイ用ビデオ・エンコーダ
- 各種インターフェース(SDカード，ATA，ATAPI，USB，コンパクト・フラッシュ)の増設カード

本システムは，ソフトウェア・ベースのプラットホームであるため，さまざまな目的に使用できます．例えば，ラピッド・プロトタイピングに使用できます．また，これと並行してカスタマイズしたハードウェアの開発を始めることも可能です．

本システムは，ソフトウェア(デモ版)も含めて2005年末までに提供される予定です．



写真A-1　低価格帯ナビゲーション・システムのリファレンス設計

ジオ・システム(SPORT送信ピンに接続されている)を利用してこうした変更が通知されます．

● 部品を追加して製品の差異化を図る

製品を差異化するため，**図2**のシステムにいくつかの要素を追加することができます．例えば，GSM(global system for mobile communication)モジュールを追加すれば，既存のマイクロホンを利用して，ハンズフリー・システムを構築できます．また，電話の使用状況に合わせて，ラジオとナビゲーションのメッセージの音量を制御することも可能です．あるいは，UARTやSPORTにBluetoothモジュールを接続して，ヘッドセット付きのハンズフリー・システムとして使用するという方法もあります．

SDカードやコンパクト・フラッシュ，USBメモリ，ハード・ディスク装置など，さまざまな記録媒体を接続してもよいでしょう．取り外し可能な記録媒体を使用すれば，事前に編集されたさまざまな地図を利用できます．記録媒体には，かならずしも電子地図を格納する必要はありません．例えば，MP3形式などで符号化したオーディオ・データを格納し，ジュークボックスとして機能させることもできます．

将来，自動車にはさまざまな運転者支援システムが搭載されるでしょう注3．こうしたシステムの一つとして，例えばCANバス経由で接続したカメラを使って道路標識を検出するものが挙げられます．カメラは道路標識を検出するたびにCANバス上にメッセージをブロードキャスト(送信)します．標識の内容がディスプレイに表示されるだけでなく，例えば運転者が標識に示された制限速度を超えた場合に警告音を発するといったことも考えられます．こうしたシステムでは，制限速度や一方通行路など，電子地図に関連する道路の状況が変化した場合，これをフラッシュ・メモリに格納し，今後の経路計算に役立てることができるでしょう．

なお，プロセッサ・ベースのプラットホームでは，アプリケーションの追加・変更はソフトウェアによって行われるので，ハードウェアを変更する必要はほとんどありません．

2. プラットホームとしてのソフトウェア

図3に，**図2**のハードウェアを用いたナビゲーション・システムにおけるソフトウェア・モジュールの構成を示します．それぞれのモジュールを個別に注目するのではなく，エンド・ユーザの視点でソフトウェアを確認できるようにしています(p.109のコラム「カー・ナビゲーション・システムにおけるソフトウェア処理」を参照)．これは，モジュールどうしがどのような相互作用を及ぼしているのかを把握するときにも役立ちます．

注3：現在，路面の状況に応じて自動車の姿勢を安定させたり，車線を維持して走行したり，前方の自動車の走行に合わせて速度を調整するような運転者支援システムが搭載されている車種もある．

● CPUパワーを考慮して実行するタスクを割り振る

システムにおいて，搭載されている機能によってはお互いに排除し合うケースが生じます．単純なジュークボックスの例を取り上げると，CDデータを読み取って符号化し，フラッシュ・メモリに格納するという作業には，かなりのCPUパワーを必要とします．このとき，CDプレーヤはすでにジュークボックス処理に占有されているため，この時点でナビゲーション機能を実行することは不可能です．ただし，CDデータの符号化が終了すると直ちに，それまで占有されていたCPUパワーを音楽再生とナビゲーションに利用するようにします．**表1**に，ナビゲーション機能とオーディオ機能についてのタスクを組み合わせた例を示しま

表1　ナビゲーションとオーディオの並行処理

		ナビゲーション処理				オーディオ処理			
		目的地入力	経路計算	ガイダンス・メッセージ	測位，GPS＋推測航法	フラッシュ・メモリへのCDデータの格納	フラッシュ・メモリ再生	オーディオCD再生	MP3再生
ナビゲーション処理	目的地入力	—	×	×	○	×	○	×	×
	経路計算		—	×	○	×	○	×	×
	ガイダンス・メッセージ			—	○	○	○	○	○
	測位，GPS＋推測航法				—	○	○	○	○
オーディオ処理	フラッシュ・メモリへのCDデータの格納					—	○	×	×
	フラッシュ・メモリ再生						—	×	×
	オーディオCD再生							—	×
	MP3再生								—

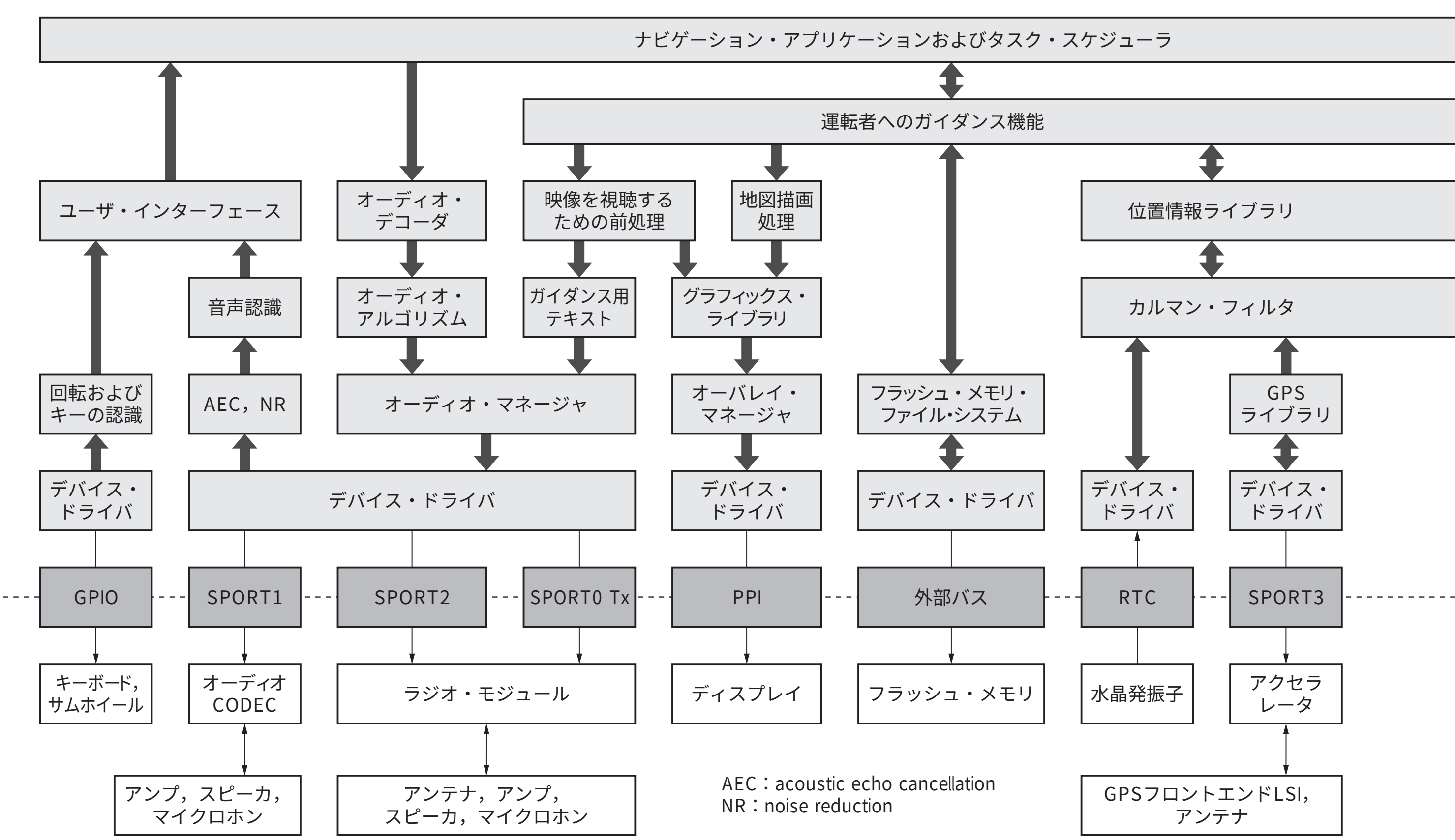


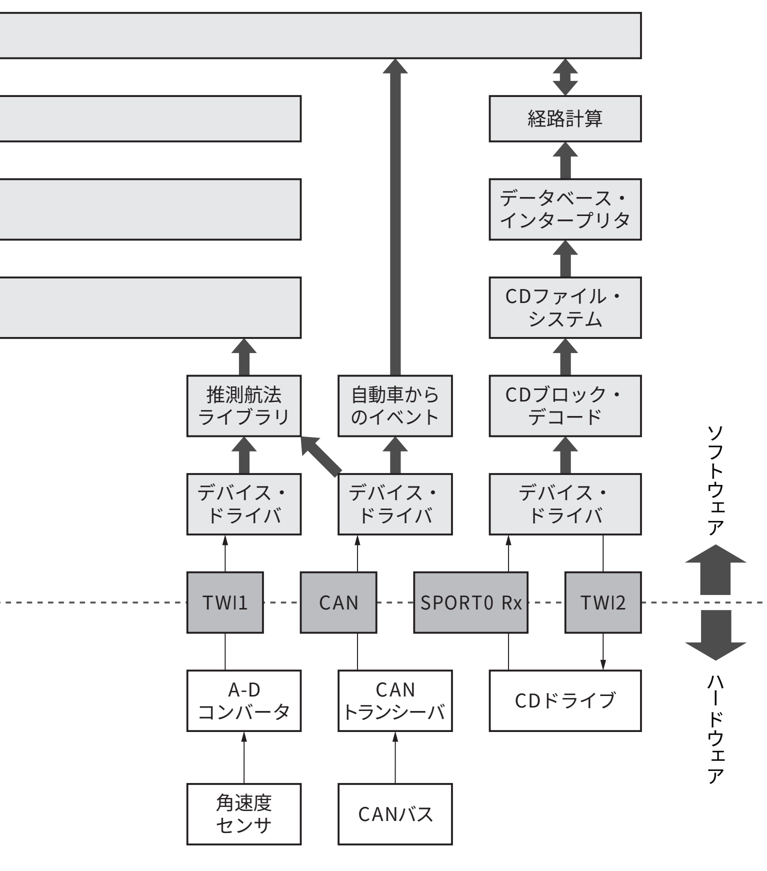
図3　ソフトウェアのインターフェース

図2のハードウェアにおけるソフトウェア・モジュールの構成を示す． CANバスからは自動車の状態に関する多くの情報が得られる．この情報を用いて，例えば，運転者側のドアが開いた時点でBlackfinプロセッサのCANコントローラにトリガをかけ，ナビテーション・システムの立ち上がり時間を早くすることもできる．

す．当然ですがこの組み合わせは，ナビゲーション・システムとして必要となる機能の一部にすぎません．

表1において，×印は二つのタスクが同時に実行されることはない（あるいは実行できない）ことを，○印は二つのプロセスを並行して実行できることを表しています．例えば，ユーザがまだ目的地を入力しているときに経路を計算しても意味がありません（表では×）．また，本稿で紹介しているシステムでは，MP3プレーヤやオーディオ用CDプレーヤがオーディオを再生している場合，新しい目的地や経路を計算することができません（表では×）．ただし，フラッシュ・メモリによってオーディオが再生されている場合は，ナビゲーション機能を使用できます（表では○）．

一般に，定期的なタスクは優先順位がもっとも高く，ほぼあらゆる環境下でそのタスクを実行する必要があります．これらのタスクのスケジュールを連携させてソフトウェアを設計することで，不要な状況切り替えをなくし，CPUパワーを節減できます．例えば，新しいGPSデータが利用可



Column2 カー・ナビゲーション・システムにおけるソフトウェア処理

本文の**図3**のソフトウェア・モジュールについて，もう少し詳しく説明しておきます．

通常のナビゲーション・システムでは，現在位置はフラッシュ・メモリに格納されており，これによってGPSのウォームアップ時間を短縮しています（これを「ホットスタート」と呼ぶ）．

ユーザに対するナビゲーションは，新しい目的地を入力することで開始し，都市リストや道路リストから住所を選択します．目的地はキーパッドやサムホイール・スイッチから入力され，これを復号して変更内容を読み取ります．住所リストはCDドライブから取得します．本来，ディジタル・データを読み取るためには，高価なCD-ROMモジュールが必要ですが，この代替手段として，より安価なオーディオ用CDモジュールを使用することが考えられます．CDモジュールから読み取られたデータはオーディオ・ブロックに格納されます．この場合，セクタの同期フィールドの検索，リードソロモン復号，逆スクランブル，エラー検出などの処理によるオーバヘッドが生じます（これらの処理はプロセッサで行われる．Blackfinの場合，等倍の読み取り速度では性能要件の約5％を使用）．またCDモジュールには，経路計算のための電子地図データも格納されています．

ユーザが目的地を入力すると，経路が計算されて，運転ガイダンスの基本モジュール機能によって地図が連続的に更新されます．地図は，電子地図のベクトル情報に基づいて計算され，グラフィックス・ライブラリを用いて，あるバッファに描画されます．別のバッファを使用すれば，同時にディスプレイに地図を表示できます．このとき，タイミングを合わせるためにDMA（direct memory access）を使用します．新しい地図の描画が完了すると，DMAは直ちに別のバッファに切り替わります．新しい地図を描画するには位置情報の更新が必要であり，これは1秒周期の同期プロセスによって実行されます．

位置情報ライブラリは，カルマン・フィルタ[注B-1]と呼ばれるアルゴリズムを通じて，GPSライブラリと推測航法ライブラリの重み付け結果，および履歴を取得します．このフィルタによって測定誤差を排除することで，情報の安定性が改善されます．

ユーザが経路上を走行し続けるためには，ポイントとなる場所を通過する前に，その地点について音声ガイダンスやディスプレイ表示で運転者へ知らせる必要があります．通過点までの距離が走行速度に応じて計算されたしきい値を超えるとトリガがかかり，音声メッセージが出力されます．同時に，操縦ガイダンス（ほとんどの場合，矢印の形）がグラフィックス・ライブラリに基づいて描画され，オーバレイ・マネージャによってディスプレイに配置されます．

オーディオ・マネージャは，ガイダンスが始まると，その音声がユーザに明瞭に聞こえるように，オーディオの音量を自動的に弱めます．

注B-1：物体の未来の状態を推定する再帰フィルタとして知られている．

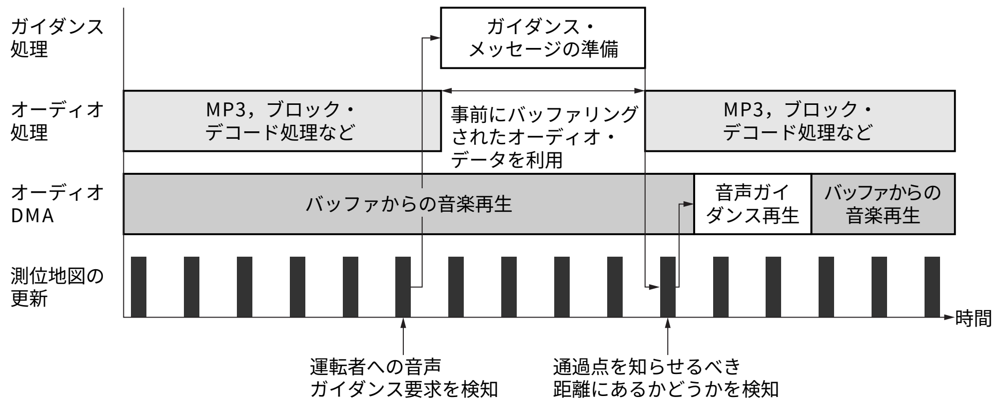


図4
動的ソフトウェアの例
ナビゲーションとオーディオ・システムが稼働しているという想定．MP3データの再生と音声ガイダンスを実行するソフトウェアが，どのようなタイミングで処理を行っているかを示す．

能になると割り込みが発生します．これらの値を処理した後，推測航法の結果が計算できるようになります．その後，システムはカルマン・フィルタを実行させ，地図上の新しい位置を推定し，新しい地図を描画します．次にディスプレイ用のバッファが切り替えられ，運転者ガイダンスの条件がすべてテストされます．すべてのタスクは連続して実行されるため，データ駆動イベントはプリエンプティブな状況切り替えを1回だけ要求します．

非定期的なタスクのスケジュールを設定する場合の例を**図4**に示します．図のいちばん下の黒色で示したスケジュールでは，上述のデータ駆動イベントが行われます．ここでは，ナビゲーションとオーディオ・システムが稼働していることを想定しています．ナビゲーション・システムは，フラッシュ・メモリに格納してある事前に計算された経路から必要な情報を取得し，オーディオ・システムはCDデータの再生を実行しています．この例では，ほとんどの時間はバッファに対してMP3復号が行われていますが，周期的な測位タスクによって定期的に割り込みがかかります．

図4の例では，ガイダンス実行用ソフトウェアは6回目のデータ駆動イベントまたはGPS駆動イベントによって，進行方法に関するメッセージを運転者のために準備する必要があることを認識します．すると，MP3復号とその関連タスクは停止しますが，オーディオの生データが事前にバッファリングされているため，再生はDMA経由で継続されています．この時点では，CPUパワーはガイダンス・メッセージの準備に使用されます．準備が完了すると，直ちに元のタスク（すなわちMP3復号処理）が続行され，音声ガイダンスを再生します．

● アセンブリ・コードでアルゴリズムの効率を確認する

現在の組み込みソフトウェアの多くは，特定プロセッサの拡張機能に対応したANSI-Cコードを利用して開発されています．また，CPUパワーを節約するためにアセンブリ言語を用いて最適化することがあります．この方法はある程度の労力が必要となりますが，その反面，プロセッサで実行されるアルゴリズムの効率を把握するうえで役立ちます．例として，CDブロックの復号時に使用する巡回冗長コード（CRC）検査サブモジュールのアセンブリ・コードを**リスト1**に示します．

リスト1はCコンパイラに合わせて記述されています．この例は，プロローグ（A）とエピローグ（E）で構成されています．スタックを介してレジスタを保存（save）および復元（restore）するという命令を使用しています．下位レジスタ（スクラッチ・レジスタ）は実行環境によって自動的に保存されます．

レジスタR0～R2には関数の引き数が格納されます．CRCアルゴリズムは特定の多項式を用いて入力ビットを除算することによって余りを算出します．実装例としては，事前に計算された参照テーブルを用いて，CRCの余りを格納したレジスタに続く八つの状態を求めます．ループの前後に八つの命令（**リスト1**のBとDの部分）を挿入することで，その八つの状態を求めることができます．ループ内にも同じ命令が記述されていますが，依存性を防止するために再度並べ替えて最適化しています．**リスト1**のCの部分で並列命令が発行されていることがわかります．これは，値の処理と読み込みの組み合わせを表しています．最終的にCRCの結果は，レジスタR0に戻り値として保存されます．

3．消費電力算出における“車載”特有の留意点

システムのボード設計を開始する前に，消費電力の推定

リスト1　32ビット巡回冗長コード・サブモジュールのアセンブリ・コード

```
// crc32.asm
// unsigned char crc32(unsigned long *inout,
// unsigned long *table, unsigned short length);
// Look-up table contains 256*32bit words.

.section program;
_crc32:
  // Prologue:
  // ---------------------------------------       ┐
  LINK 8;                                          │
  [--SP]=(R7:4, P5:3);                             │
  // save preserved registers R7:R4, P5:3          ├ A
  SP += -8;                                        │
  // ---------------------------------------       ┘
  P2 = R2;           // R2 contains length in bytes
  P0 = R0;           // R0 points to buffer and
                     // contains finally the result
  P1 = R1;           // R1 points to CRC-table
  R0 = 0;            // initial CRC-value
  P3 = 1;            // set P3 to 1
  P2 -= P3;          // decrement loop with 1
  R3 = B[P0++] (Z);  // (1) load buffer-value         ┐
  R0 = R0 ^ R3;      // (2) XOR result, buffer-value  │
  R2 = R0.b (Z);     // (3) mask last 8 bits (copy byte) ├ B
  P3 = R2;           // (4) copy to pointer           ┘
  R6.l = 0x88;       // information for
                                    extracting 8 LSBs
  LSETUP( .begin_loop, .end_loop) lc0 = P2;

.begin_loop:
  R1 = R0 >> 8 || R3 = B[P0++] (Z);// (5)&(1)            ┐
                 //shift result 8 bits, load input       │
  R0 = R1 ^ R3;   // (8) XOR                             │
                 //Shifted CRC-Status,32-bit XOR-result  │
  P3 = P1 + (P3<<2); // (6)                              ├ C
               //calculate Base + [4*bytes]              │
                                         LUT-position    │
  R1 = EXTRACT (R0, R6.l) (Z) || R3 = [P3]; // (7)       │
               // get LUT-value                          ┘
  R4 = R3.b (Z); // (3) Mask last 8 bits (copy byte)
  R2 = R1 ^ R4;  // (2) XOR (CRC-Status,
                                      LUT-value) 8-bit
  P3 = R2;       // (4) Copy 8-bit XOR-result to pointer
.end_loop:     R0 = R0 ^ R3; // (2)
               //XOR (CRC-Status, LUT-value) 32-bit

  NOP;
  NOP;
  R1 = R0 >> 8;  // (5) shift current result 8 bits left  ┐
  P3 = P1 + (P3<<2); // (6) calculate base + 4 * value    ├ D
  R3 = [P3];     // (7) get LUT-value                     │
  R0 = R1 ^ R3;  // (8) XOR result, LUT-value             ┘

  // Epilogue:
  // ----------------------------------------   ┐
  SP += 8;                                      │
  (R7:4, P5:3)=[SP++];                          │
  // restore preserved registers R7:R4, P5:3    ├ E
  UNLINK;                                       │
  // ----------------------------------------   ┘

  RTS;
._crc32.end:
.global _crc32;
```

値を計算することが重要です．自動車業界では，情報系システムに対する動作温度範囲の要件として最大85 ℃と規定されています．このことは，電源電流の要件も規定されることを意味します．

以下に示す例では，標準的なアプリケーションがDSP（ここではBlackfinプロセッサ）で実行されていることを想定して消費電力を計算しています[注4]．

● 温度依存の静的電流と温度に依存しない動的電流

Blackfinプロセッサには二つの電圧領域があります．一つは内部論理回路（コア）電圧で，もう一つはI/O電圧です．二つの積和演算回路を75 %，加算演算回路を25%実行している標準的なアプリケーションでは，消費電流の測定値は次のようになります[(1)]．

$$I_{DDTYP_DYN(500MHz)} = I_{DDTYP(500MHz)} - I_{DEEPSLEEP(25C)}$$
$$= 190 - 31 = 159\ [\mathrm{mA}]$$

ここで，$I_{DDTYP_DYN(500MHz)}$は500MHz動作時の動的電流，
$I_{DDTYP(500MHz)}$は500MHz動作時の消費電流，
$I_{DEEPSLEEP(25C)}$は動作温度25 ℃における静的電流

なお，動的電流（論理回路のスイッチングの際に流れる電流）は温度に依存しません．

次に，85 ℃における静的電流（待機時の消費電流）は，次式で推定することができます．

$$I_{DEEPSLEEP(85C)} = I_{DEEPSLEEP(85C)} \times e^{0.015(T1-T0)}$$
$$= 31e^{0.015(85-25)} = 76.24\ [\mathrm{mA}]$$

よって，85 ℃におけるデバイスの消費電力は以下のようになります．

$$P_{DDINT} = V_{DDINT}(I_{DEEPSLEEP(85C)} + I_{DDTYP_DYN(500MHz)})$$
$$= 1.2 \times (76.24 + 159) \quad 282\ [\mathrm{mW}]$$

トランジスタのリーク電流を含めたとしても，85 ℃では1サイクル当たり564pW（＝282mW ÷ 500MHz）となります．

本プロセッサにはオンチップ電圧レギュレータが搭載されています．この電圧レギュレータは，I/Oピン用として使われる外部電圧を内部電圧に変換します．変換効率は約75 %です．そのため，プロセッサ内部の総消費電力についての要件は約376mW（=282mW÷0.75）となります．外部の消費電力はおもに能動部品によって決まり，静的電流は無視されます．すなわち，プリント基板上における消費電力は，スイッチング周波数，負荷容量，トグル比，稼働率，

注4：デバイスの動的電流および静的電流の詳細は参考文献(1)を参照のこと．また，周辺機器用のプリント基板上で電流を駆動するI/Oドライバの決定については参考文献(2)を，対応する外部電源については参考文献(3)を参照のこと．

表2　消費電力(3.3Vにおける推定値)

周辺回路	周波数(Hz)	出力ピン数	負荷容量(F)	トグル比	稼働率	外部電圧 V_{DD}(V)	消費電力(mW)
CAN	1M	1	30p	1	0.05	3.30	0.01
SPORT0 Tx オーディオ	1.411M	3	30p	1	1.00	3.30	0.69
SPORT1 Tx オーディオ	1.411M	3	30p	1	1.00	3.30	0.69
SPORT2 Tx オーディオ	1.411M	3	30p	1	1.00	3.30	0.69
SPORT3 Tx GPS	40M	3	30p	1	0.50	3.30	9.80
PPI	27M	16	30p	1	0.25	3.30	17.64
TWI0 角速度センサ	100k	2	30p	1	0.25	3.30	0.01
TWI1(I^2C)	100k	2	30p	1	0.25	3.30	0.01
UART デバッグ	115k	2	30p	1	0.00	3.30	0.00
SPIブート	20M	3	30p	0.5	0.00	3.30	0.00
SDRAM/NAND	133M	36	30p	0.25	0.75	3.30	146.65
							計176.19

動作電圧などをもとに，次式で決定します[2]．

$$P_{DDEXT} = V_{DDETX}^2 \times C_L \times f/2 \times O \times TR \times U$$

ここで，P_{DDEXT}はプリント基板上の消費電力，V_{DDETX}は外部電圧，C_Lは負荷容量，fは動作周波数，Oは出力ピン数，TRはトグル比，Uは稼働率

この式から導き出した結果を**表2**に示します．85 ℃におけるシステムとしての総消費電力は，デバイス内部の消費電力376mWとプリント基板上の消費電力176mWを合わせた552mWになります．ほかの構成要素や安全マージンを加味すると，3.3Vレールのレギュレータ回路を規定することができます．

● パワーダウン時のトラフィック・メッセージの捕捉

駐車中にナビゲーション・システムが起動されていない場合，RDS/TMCメッセージは受信も保存もされません．反面，ナビゲーション・システムは消費電力が多すぎるため，常時起動しておくことはできません．

駐車時のメッセージを保存するためには，リアルタイム・クロックでプロセッサを起動し，低いデューティ比でシステムの一部だけを稼働させることによって消費電力を低減するという方法を用いることができます．このとき，待機状態の本プロセッサはリアルタイム・クロックに電力を供給するために3.3Vで170μWを消費することになります[1]．プロセッサはときどき稼働して，ラジオのトラフィック・メッセージを受信します．こうすれば，プロセッサはCPUパワーを100％使用する必要がなく，消費電力を節約することができます．

この目的のため，内部電圧が0.8Vで50MHz動作のオンチップ電圧レギュレータを使用しています．アプリケーション・ソフトウェアごとにプロセッサを設定する個別のカーネルでブートすることにより，稼働中のプロセッサのCPUパワーを低減しています．このとき，プロセッサの内部回路は49.5mWを消費しますが，外部電力は無視できます[1]．駐車時の平均消費電力は，デューティ比δ(ほぼ24時間の平均値)によって決まります．これは稼働時間を表すものであり，以下の式で計算されます．

$$P_{DDAVE} = \delta \times P_{DDINT} + (\delta - 1) \times P_{DDRTC}$$

ここで，P_{DDAVE}は平均消費電力，P_{DDINT}はプロセッサ内部の消費電力，P_{DDRTC}はリアルタイム・クロックで消費される電力

デューティ比が1％の場合，駐車時のプロセッサの平均消費電力は330μWになります．デューティ比はトラフィック・メッセージの取り込みにかかる時間だけでなく，メッセージが利用可能かどうかにも左右されます．RDSブロードキャストが「利用可能なメッセージがない」と通知した場合，プロセッサは直ちに元の待機状態に移行します．

参考・引用*文献

(1) Analog Devices；ADSP-BF531/ADSP-BF532/ADSP-BF533 Blackfin Embedded Processor Data Sheet，Rev. B，June 2005.
(2) Joe Beauchemin；Estimating Power for ADSP-BF533 Blackfin Processors，Application note，Rev. 1，Analog Devices，February 2004.
(3) Analog Devices；External Power Spreadsheet. Associated file with Estimating Power for ADSP-BF533 Blackfin Processors (EE-229)，February 2004.

Paffen Matthijs
Analog Devices社　System Engineer

<筆者プロフィール>
Paffen Matthijs．2002年にオランダのEindhoven工科大学電気工学科の理学修士号をとる．その後，組み込み分野におけるDSPアプリケーションを中心としたシステム開発に従事．今日，ナビゲーション・システムなどの車載情報機器の開発を行っている．